Programming Fundamentals
Lecture #16
Formatted Input/Output in C Programming

Junaid Hassan

Lecturer CS & IT Department UOS MBDIN

junaidte14@gmail.com

- Many features of printf and scanf were discussed earlier
- This lecture summarizes those features and introduces others
- C provides standard functions scanf() and printf(), for performing formatted input and output .These functions accept, as parameters, a format specification string (e.g %d) and a list of variables

- All input and output is performed with streams, which are sequences of bytes.
- In input operations, the bytes flow from a device (e.g., a keyboard, a disk drive, a network connection) to main memory.
- In output operations, bytes flow from main memory to a device (e.g., a display screen, a printer, a disk drive, a network connection, and so on)

- When program execution begins, three streams are connected to the program automatically
- Normally, the standard input stream is connected to the keyboard and the standard output stream is connected to the screen
- A third stream, the standard error stream, is connected to the screen

- Precise output formatting is accomplished with printf
- Every printf call contains a format control string that describes the output format and other-arguments (which are optional) correspond to each conversion specification in format-control-string
- Each conversion specification begins with a percent sign and ends with a conversion specifier
- There can be many conversion specifications in one format control string

Printing Integers

```
3  #include <stdio.h>
4
5  int main( void )
6  {
7     printf( "%d\n", 455 );
8     printf( "%i\n", 455 ); /* i same as d in printf */
9     printf( "%d\n", +455 );
10    printf( "%d\n", -455 );
11    printf( "%hd\n", 32000 );
12    printf( "%ld\n", 2000000000L ); /* L suffix makes literal a long */
13    printf( "%o\n", 455 );
14    printf( "%u\n", 455 );
15    printf( "%u\n", -455 );
16    printf( "%x\n", 455 );
17    printf( "%X\n", 455 );
18    return 0; /* indicates successful termination */
19 } /* end main */
```

```
455
455
455
-455
32000
2000000000
707
455
4294966841
1c7
1C7
```

Printing Floating Point Numbers

```c
/* Fig 9.4: fig09_04.c */
/* Printing floating-point numbers with
   floating-point conversion specifiers */

#include <stdio.h>

int main( void )
{
   printf( "%e\n", 1234567.89 );
   printf( "%e\n", +1234567.89 );
   printf( "%e\n", -1234567.89 );
   printf( "%E\n", 1234567.89 );
   printf( "%f\n", 1234567.89 );
   printf( "%g\n", 1234567.89 );
   printf( "%G\n", 1234567.89 );
   return 0; /* indicates successful termination */
} /* end main */
```

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

Printing Characters and Strings

```c
/* Fig 9.5: fig09_05c */
/* Printing strings and characters */
#include <stdio.h>

int main( void )
{
   char character = 'A'; /* initialize char */
   char string[] = "This is a string"; /* initialize char array */
   const char *stringPtr = "This is also a string"; /* char pointer */

   printf( "%c\n", character );
   printf( "%s\n", "This is a string" );
   printf( "%s\n", string );
   printf( "%s\n", stringPtr );
   return 0; /* indicates successful termination */
} /* end main */
```

```
A
This is a string
This is a string
This is also a string
```

Other Conversion Specifiers

```
2  /* using the p, n, and % conversion specifiers */
3  #include <stdio.h>
4
5  int main( void )
6  {
7     int *ptr; /* define pointer to int */
8     int x = 12345; /* initialize int x */
9     int y; /* define int y */
10
11    ptr = &x; /* assign address of x to ptr */
12    printf( "The value of ptr is %p\n", ptr );
13    printf( "The address of x is %p\n\n", &x );
14
15    printf( "Total characters printed on this line:%n", &y );
16    printf( " %d\n\n", y );
17
18    y = printf( "This line has 28 characters\n" );
19    printf( "%d characters were printed\n\n", y );
20
21    printf( "Printing a %% in a format control string\n" );
22    return 0; /* indicates successful termination */
23 } /* end main */
```

```
The value of ptr is 0012FF78
The address of x is 0012FF78

Total characters printed on this line: 38

This line has 28 characters
28 characters were printed

Printing a % in a format control string
```

Printing with Field Widths and Precision

```c
/* Fig 9.8: fig09_08.c */
/* Printing integers right-justified */
#include <stdio.h>

int main( void )
{
   printf( "%4d\n", 1 );
   printf( "%4d\n", 12 );
   printf( "%4d\n", 123 );
   printf( "%4d\n", 1234 );
   printf( "%4d\n\n", 12345 );

   printf( "%4d\n", -1 );
   printf( "%4d\n", -12 );
   printf( "%4d\n", -123 );
   printf( "%4d\n", -1234 );
   printf( "%4d\n", -12345 );
   return 0; /* indicates successful termination */
} /* end main */
```

```
   1
  12
 123
1234
12345

  -1
 -12
-123
-1234
-12345
```

Printing with Field Widths and Precision

```c
/* Using precision while printing integers,
   floating-point numbers, and strings */
#include <stdio.h>

int main( void )
{
   int i = 873; /* initialize int i */
   double f = 123.94536; /* initialize double f */
   char s[] = "Happy Birthday"; /* initialize char array s */

   printf( "Using precision for integers\n" );
   printf( "\t%.4d\n\t%.9d\n\n", i, i );

   printf( "Using precision for floating-point numbers\n" );
   printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );

   printf( "Using precision for strings\n" );
   printf( "\t%.11s\n", s );
   return 0; /* indicates successful termination */
} /* end main */
```

```
Using precision for integers
        0873
        000000873

Using precision for floating-point numbers
        123.945
        1.239e+002
        124

Using precision for strings
        Happy Birth
```